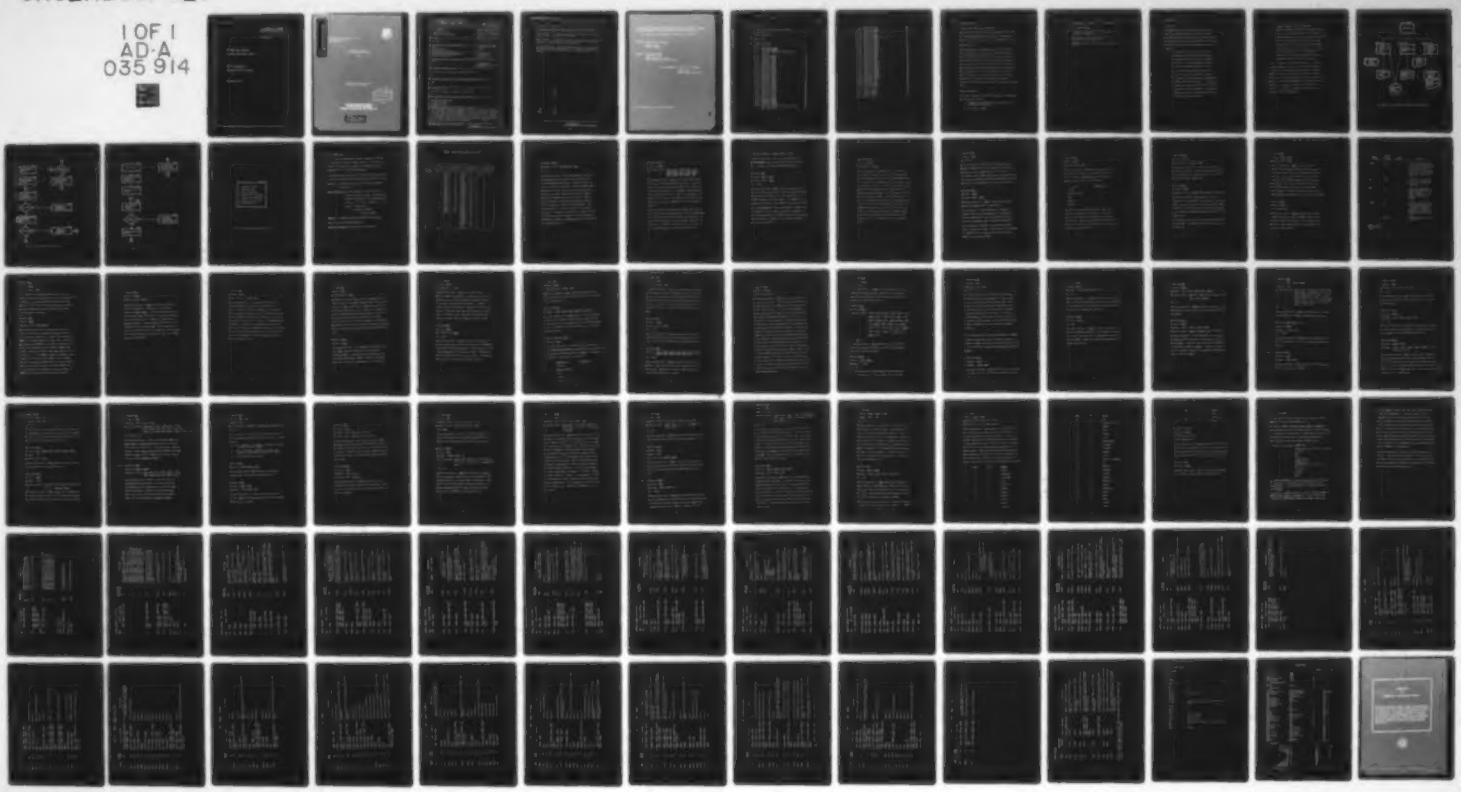
AD-A035 914 TRW INC REDONDO BEACH CALIF F/G 9/2
FORTRAN CODE AUDITOR - PROGRAM MAINTENANCE MANUAL.(U)
DEC 76 P SMITH F30602-76-C-0187
UNCLASSIFIED RADC-TR-76-395-VOL-2 NL

1 OF 1
AD-A
035 914



END
DATE
FILMED
3-25-77
NTIS

U.S. DEPARTMENT OF COMMERCE
National Technical Information Service

AD-A035 914

FORTRAN CODE AUDITOR

PROGRAM MAINTENANCE MANUAL

TRW, INCORPORATED
REDONDO BEACH, CALIFORNIA

DECEMBER 1976

AD A035914

RADC-TR-76-395, Volume II (of two)
Final Technical Report
December 1976

FORTRAN CODE AUDITOR
Program Maintenance Manual

TRW

Approved for public release;
distribution unlimited.

DDC
RECEIVED
FEB 22 1977
A

ROME AIR DEVELOPMENT CENTER
AIR FORCE SYSTEMS COMMAND
GRIFFISS AIR FORCE BASE, NEW YORK 13441

REPRODUCED BY
NATIONAL TECHNICAL
INFORMATION SERVICE
U. S. DEPARTMENT OF COMMERCE
SPRINGFIELD, VA. 22161

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-76-395, Vol II (of two)	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) FORTRAN CODE AUDITOR Program Maintenance Manual		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report May 1975 - January 1976
		6. PERFORMING ORG. REPORT NUMBER N/A
7. AUTHOR(s) Paul Smith		8. CONTRACT OR GRANT NUMBER(s) F30602-76-C-0187
9. PERFORMING ORGANIZATION NAME AND ADDRESS TRW One Space Park Redondo Beach CA 90278		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 55811409
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (ISIS) Griffiss AFB NY 13441		12. REPORT DATE December 1976
		13. NUMBER OF PAGES 82
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Same		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Donald L. Mark (ISIS)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

FORTRAN Coding Standard
Structural Analysis
Conventions Audit
Automatic Test Tool

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The FORTRAN Code Auditor, an automated test tool, is used for the cost effective enforcement of FORTRAN programming standards and conventions appropriate to the Air Force software environment. It does not modify code. Using predefined coding standards and conventions, it simply advises the user where these standards and conventions have not been adhered to. The major advantage of favoring an automated auditor over manual methods, besides cost effectiveness, is complete objectivity and unambiguity. (over)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

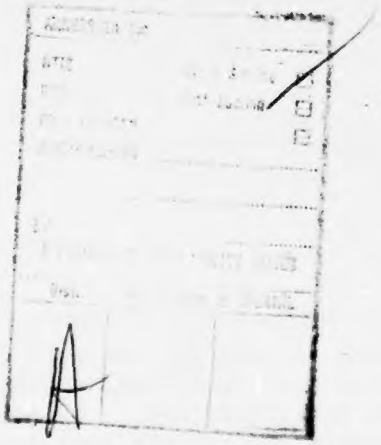
The standards can be viewed as being coding enforcements in four areas:

1. Documentation Standards - Standards defining quantity and placement of commentary thus enhancing program readability and comprehensive.

2. Format Standards - Standards identifying physical placement and grouping of code elements on the source code listing.

3. Design Standards - Standards limiting module size and placing restrictions on the use of certain instructions with the end result of providing an optimization of code relative to execution time.

4. Structural Standards - Standards requiring the use of strict rules for the top-down design and implementation of a system of programs and the requirement that the components adhere to a hierarchical form as much as possible.



This report has been reviewed by the RADC Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public including foreign nations.

This report has been reviewed and is approved for publication.

APPROVED: Ronald L. Mark

RONALD L. MARK
Project Engineer

APPROVED: Robert D. Kreutz

ROBERT D. KREUTZ, Col, USAF
Chief, Information Sciences Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

Do not return this copy. Retain or destroy.

CONTENTS

CONTENTS (Continued)

1.0 GENERAL DESCRIPTION

1.1 Purpose of the Program Maintenance Manual

The objective of the Program Maintenance Manual is to provide the maintenance programmer personnel with the information necessary to effectively maintain the RADC FORTRAN Code Auditor.

1.2 System Application

The automated test tool, Code Auditor, is used for the cost effective enforcement of FORTRAN programming standards and conventions appropriate to the Air Force software environment. It does not modify code. Using pre-defined coding standards and conventions, it simply advises the user where these standards and conventions have not been adhered to in an objective and unambiguous manner. The RADC FORTRAN Code Auditor also monitors the user's source code to determine if the code is structured (i.e., consist of combinations of control structures) such that control flows from top to bottom, beginning to end.

1.3 System Configuration

The minimum configuration required to support the RADC FORTRAN Code Auditor shall include:

1. Honeywell Series 600/6000 Information Processing System under GCOS control
2. 40 K words of memory

3. DSS180 Disk Storage Subsystem or the equivalent; a minimum of 1 drive

4. 36 - bit word length

5. Card Reader - Reads punched 80 column cards (CRZ-201 or equivalent)

6. Line Printer - Prints 132 columns per line (PRT 300/PRT 201 or equivalent)

7. Software - FORTRAN Y compiler (Honeywell Extended Compiler)

2.0 SYSTEM DESCRIPTION

2.1 General Description

The Code Auditor consists of three physical overlays which are functionally consistent with the three basic logical components of the system and their attendent operation. The operations performed by the Code Auditor's logical parts and their correlation to the appropriate physical system overlays are:

A. Analyze the user's source code to check for non-adherence to the established standards and conventions, reporting any non-conformance monitored. Overlay A performs this function completely independent of Overlays B and C.

B. Overlay B performs a second pass analysis of the user's source code, parsing its structure into segments and relationships among segments for subsequent processing by Overlay C. The Segment Transfer Table, which describes this interrelationship among the program's segments, is written to a temporary disk file and constitutes the single interface between Overlays B and C.

C. Overlay C accepts as input the temporary file created in Overlay B containing the Segment Transfer Table. Via iterative application of the reduction algorithm, an analysis is made of the Segment Transfer Table to establish compliance with the structured programming rules. This overlay prints the Segment Transfer Table as passed from Overlay B, its contents after iterative reduction if unstructured and summarily a "structured/unstructured" message for each module of the user's source processed.

Figure 2.1-I depicts for each logical segment, its interface with other segments, files processed, and reports generated. The main driver (root segment) is interfaced via a labeled "COMMON" to each of the three overlays comprising the Code Auditor system. Besides providing root segment access to Code Auditor files, the labeled "COMMON" block passes option card parameters for execution control of the individual overlays. Figures 2.1-II through 2.1-IV depict functional flows of Overlays A, B, and C respectively.

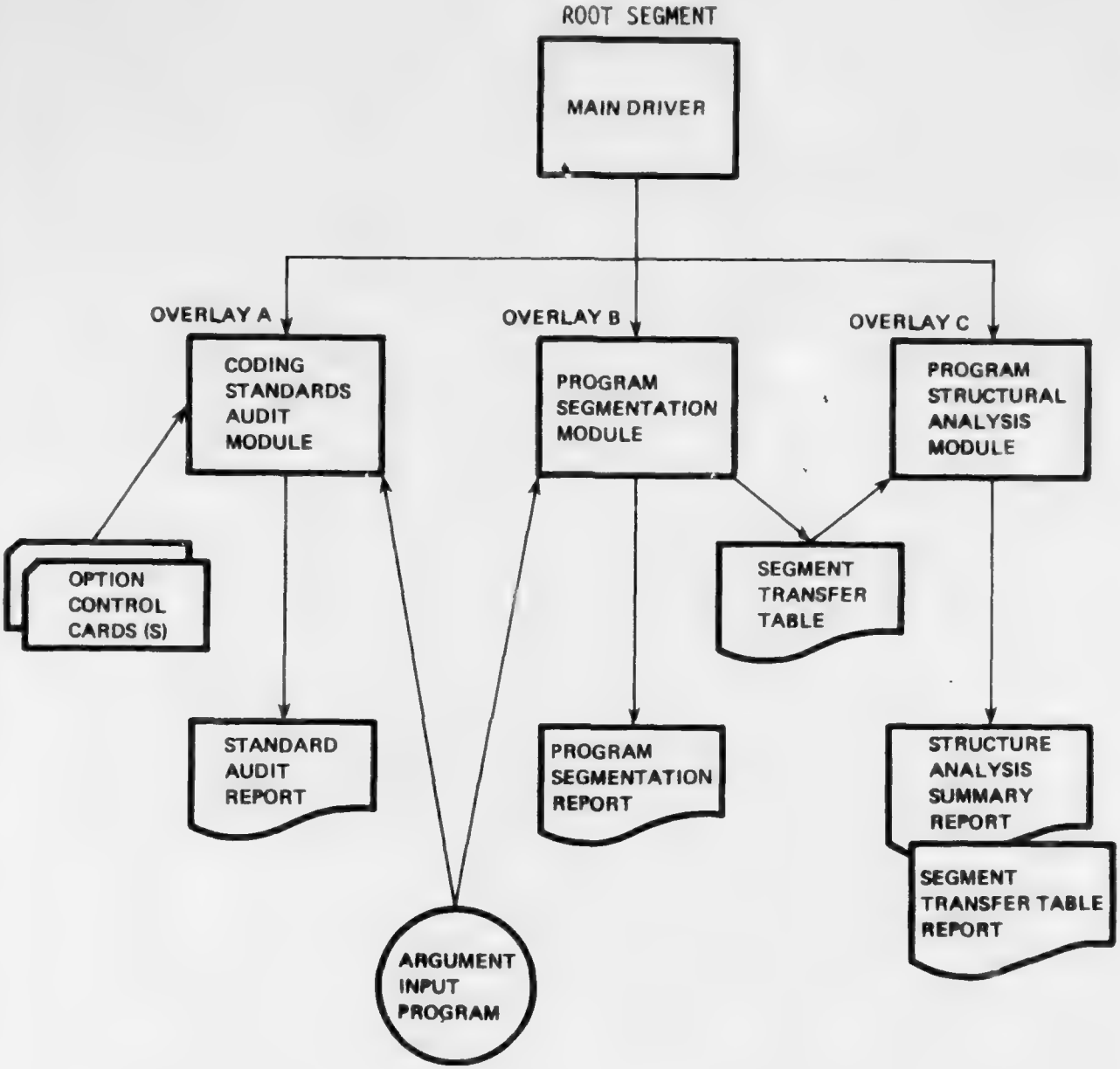


Figure 2.1-I. Code Auditor Organization and Interfaces

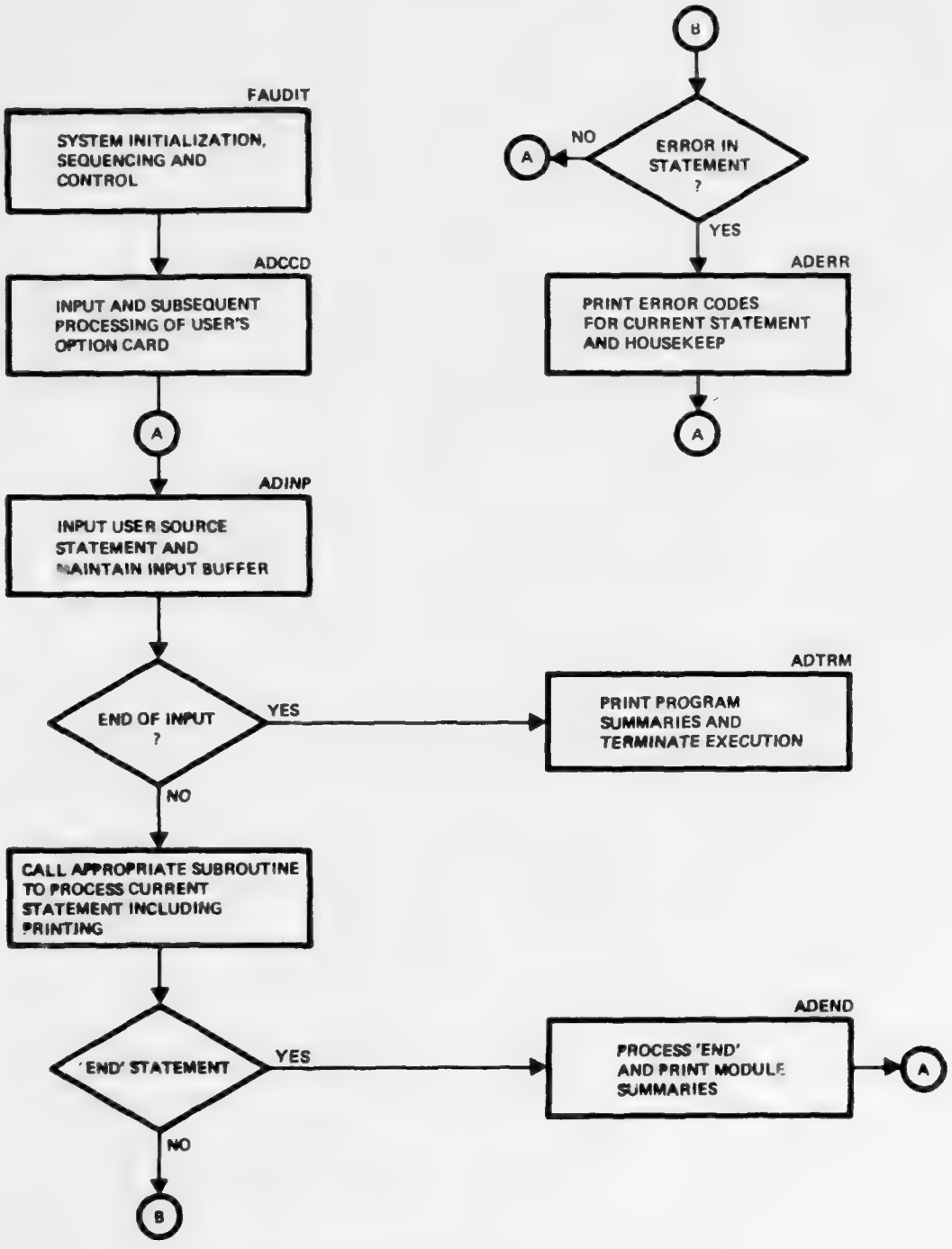


Figure 2.1-II. Overlay A Functional Flow

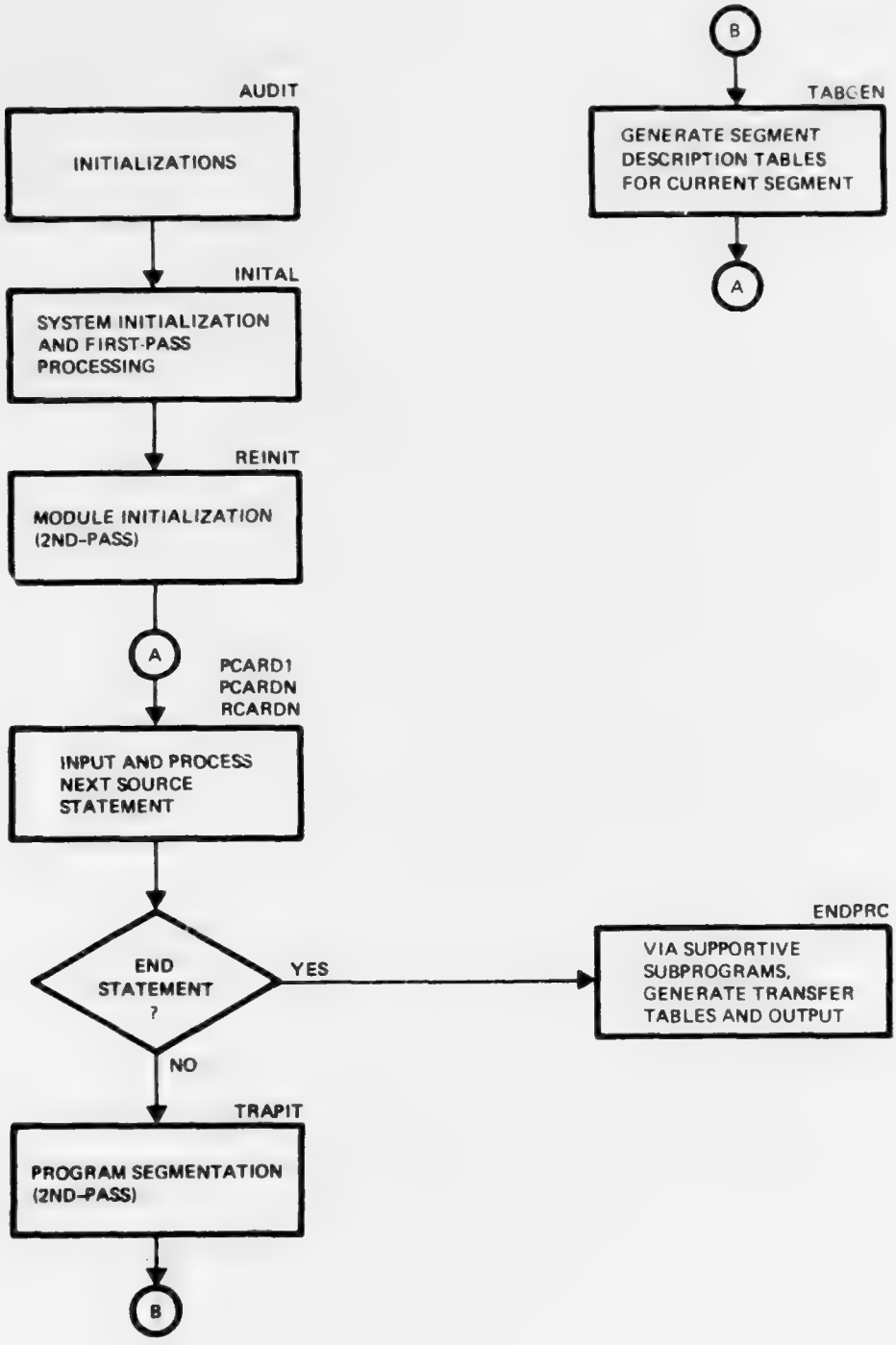


Figure 2.1-III. Overlay B Functional Flow

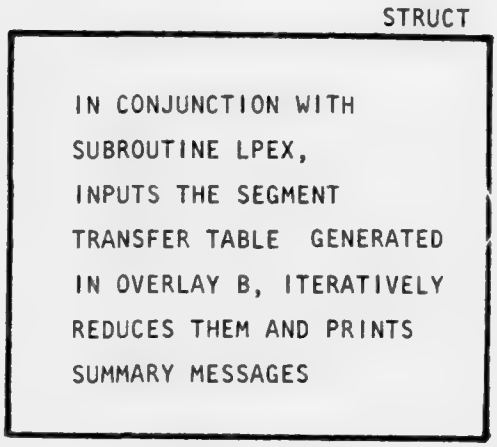


FIGURE 2.1-IV. OVERLAY C FUNCTIONAL FLOW

2.2 Detail Description

Table I lists all modules which comprise the RADC Code Auditor. It relates them to the Root Segment, Overlay A, Overlay B, and Overlay C. Each module is indexed to the program description (paragraphs 2.2.1 - 2.2.66) by page numbers.

The program description consists of five possible categories which indicate the functional characteristics of each module. They are:

Called by: Indicates module(s) which call the subject module into execution.

Subroutines called: Indicates module(s) which are called into execution by subject module. The subroutines called are presented in the following manner:

X1 (Y1, Y2, ..., Yn); ...; Xm (Y1, c, Y2, ..., Yn)

where X = module called,

Y = argument variable(s),

c = constant argument

Parameters: Parameter arguments accepted by subject module

Files: Logical Unit Variable/Number(s)

Functional Description: Description of subject module.

TABLE I Module Detail Description Index

Root Segment	Functional Description Page No.	Overlay A	Functional Description Page No.	Overlay B	Functional Description Page No.	Overlay C	Functional Description Page No.
MAINLD	13	FAUDIT	14	AUDIT	30	STRUCT	47
		ADCOD	15	ADJUST	30	LPEX	47
		ADCOD1	15	ASSIGN	31		
		FCA	15	BLK1	31		
		ADDAT	15	BLK2	31		
		ADIFS	16	BLK3	31		
		ADEND	17	BUILD	31		
		AUGOT	17	CLEROP	32		
		ADINP	18	DOPERS	32		
		ADIOE	19	SBXPTG	33		
		ADMIX	19	STLBL	33		
		ADRAY	20	TABGEN	34		
		ADCCD	20	TAPWRT	34		
		ADCON	22	UNBILD	34		
		ADCOL	22	TYPCVT	35		
		ADDOL	23	INITSG	35		
		ADRY1	24	REREAD	35		
		ADRY2	25	POINTR	36		
		ADRY3	25	BLK4	31		
		ADSTA	26	NEWSTR	36		
		ADSTR	26	IFTRAP	36		
		ARGU	27	PASST	37		
		GETWRD	27	STNTBL	37		
		NTYPE	27	TRAPIT	38		
		EXPCN	28	WRWD	38		
		TYPES	28	KTGEN	38		
		ADERR	28	KT2WD	39		
		ADINP	29	DOTERM	39		
		ADTRM	30	INITAL	40		
				PCARDI	40		
				PCARDN	41		
				REINIT	42		
				TERMIN	42		
				RCARDN	42		
				ENDPRG	43		
				BLKCHR	31		
				BCDINT	43		
				IFCK	44		
				SQZB	44		
				TYPE	45		

2.2.1 Root Segment MAINLD

Subroutines called: FAUDIT; AUDIT; STRUCT

File: 06

Functional Description - This program module provides the sequencing and control function for the three overlays of the RADC FORTRAN Code Auditor. Overlay A examines the user's source code for adherence to the RADC FORTRAN coding standards and is the first overlay called into execution. Upon return of control from Overlay A's driver program, FAUDIT, the "COMMON" resident control variable, NPA, is tested to determine whether execution control is passed to the two remaining overlays of the Code Auditor. NPA, a binary variable, is set as the result of user option card input. It declares the user's desire to audit or not audit the input code for adherence to 'structured program' requirements. Additionally, MAINLD performs rewind functions for Code Auditor files.

2.2.2 Subroutine FAUDIT

Called by: MAINLD

Subroutines called: ADCCD; ADINP; ADERR; ADSTA; ADCON; GETWRD; ADSTR; ADCOL; ADDAT; ADIMP; TYPES; ADDOL; ADIOE; ADRAY; ADGOT; ADIFS; ADEND; ARGU; ADMIX; ADTRM

Functional Description - FAUDIT is the main control routine for Overlay A of the RADC FORTRAN Code Auditor. Overlay A examines user's source code for adherence to the RADC coding standards. Subsequent to calls to ADCCD, a subroutine which reads and processes the user's option card, and ADINP, which reads the user's source input, FAUDIT performs nominal syntax analysis and via calls to GETWRD extracts source statement keywords.

Upon making keyword interpretations, FAUDIT either internally or via calls to appropriate subordinate routines, checks source input statements for adherence to RADC coding standards. It should be noted that upon return from ADCCD, the option card processor, the "COMMON" resident variable NCA is tested to determine whether Overlay A processing should continue.

2.2.3 Block Data Subprograms - ADCOD, ADCOD1, and ECA

Functional Description - Data is entered into Overlay A labeled "COMMON" areas prior to Overlay A execution by the use of BLOCK DATA subprograms ADCOD, ADCOD1, and ECA.

2.2.4 Subroutine ADDAT

Called by: FAUDIT; ADCOL

Subroutine called: ADERR

File: IOUTP

Functional Description - ADDAT is called into execution to process variables contained in various declarative statements within the user's source input. If the declarative is a "COMMON" statement, the total length of the "COMMON" block is computed and returned via the "COMMON" resident variable IDIM.

For other declaratives, "REAL", "DIMENSION", "INTEGER", etc., the dimension of each variable referenced is computed, the variable name checked for previous declaration (coding standard #10), and the name, dimension, and type of each variable is stored in the array JTAB.

2.2.5 Subroutine ADIFS

Called by: FAUDIT

Subroutine called: ADRAY

Functional Description - ADIFS performs the function of isolating the 'predicate' portion of an "IF" expression from the 'consequent' portion. It does this by beginning at the leftmost left parenthesis of the expression, counting all parenthesis in a left to right scan, and flagging the appropriate right parenthesis when the number of left parentheses match the number of right parentheses. During processing, subroutine ADRAY is called to determine if a left or right parenthesis encountered is imbedded within a hollerith string; if so, the parenthesis is ignored. The "COMMON" resident variable JJ identifies the position in STATMT (an array containing the source statement image) containing the appropriate parenthesis delimiting the 'predicate' of the "IF" statement from the 'consequent' portion.

2.2.6 Subroutine ADEND

Called by: FAUDIT

File: IOUTP

Functional Description - ADEND is called into execution whenever a FORTRAN "END" card is encountered in the user's source code. Module summary reports, including error identification, are formatted and printed. Various variables and arrays are cleared or initialized in preparation for the processing of the next module of the user's source input.

2.2.7 Subroutine ADGOT

Called by: FAUDIT

Subroutine called: ADERR

Functional Description - ADGOT is called into execution whenever the keyword "GOTO" is encountered.

If the character is a left parenthesis, indicating a computed "GOTO", the indexing variable is isolated and checked to determine if its value was checked prior to its use in the computed "GOTO" statement (standard #23).

If the character is alphabetic, indicating an assigned "GOTO", and the source input is real-time code, the statement is flagged as having violated standard #28 (no assigned "GOTO's" in real-time programs).

2.2.8 Subroutine ADIMP

Called by: FAUDIT

Subroutine called: ADIMP

Functional Description - ADIMP gains execution control when "IMPLICIT" statements are encountered within the user's source code. ADIMP determines the declaration type and assigns a numeric code as follows.

TYPE	NUMERIC CODE
"INTEGER"	1
"DOUBLE PRECISION"	2
"REAL"	3
"COMPLEX"	4
"LOGICAL"	5

Upon establishing the appropriate numeric code, the remainder of the declaration statement is scanned to identify the alphabetic characters implicitly declared. The array IMP1, containing one storage location for each letter of the alphabet, has corresponding cells assigned to the numeric value established.

2.2.9 Subroutine ADIOE

Called by: FAUDIT

Subroutines called: GETWRD; ADERR

Functional Description - The logical device name for all input/output statements is examined by ADIOE to determine if the variable name is either implicitly or explicitly declared as integer (standard #22).

2.2.10 Subroutine ADMIX

Called by: FAUDIT

Subroutines called: GETWRD; ADERR; NTYPE (L1, L2); ADIFS

Functional Description - ADMIX examines arithmetic expressions for adherence to coding standard #14 (no mixed mode expressions on right hand side of an equate). ADMIX isolates all variable names in the right-hand expression and via function subprogram NTYPE determines each variables declared type.

Upon encountering two variable names with differing types, the statement is flagged indicating violation of coding standard #14.

2.2.11 Subroutine ADRAY

Called by: ADIFS; FAUDIT

Subroutine called: ADRY1

Functional Description - ADRAY is called to process character strings (variable names or hollerith strings). If the input variable ITEST has a value greater than zero, the value is used by ADRAY to select the contents of an array element from STATMT and determine if the contents are contained within a hollerith string (IHOLT=1). The array, STATMT, contains the current source statement being processed. For values of ITEST equal to zero, a call is made to ADRY1 to further process a variable or array name.

2.2.12 Subroutine ADCCD

Called by: FAUDIT

Files: INPUT, IOUTP

Functional Description - ADCCD processes the user option card input and sets or defaults "COMMON" resident flags indicating program control options. The following table depicts user input character strings, corresponding flag settings and their meanings.

INPUT CHARACTER STRING	FLAG VARIABLE NAME	FLAG SETTING	MEANING
NOLIST	JLST	0	Directs the Code Auditor to limit the source code listing to only those statements containing audit errors in addition to summary reports. The default is a full listing of all input source code (JLST=1)
REAL	IREAL	1	Indicates real-time input source code. The default is non real-time input (IREAL=0)
NOCA	NCA	1	Directs the Code Auditor to refrain from auditing any coding standards. The default condition is all coding standards are audited (NCA=0)
NOSTRUCT	NPA	1	Directs the Code Auditor to refrain from auditing for "structured program" requirements. The default condition is to conduct a "structured program" audit (NPA=0)
xx	NRES(xx)	1	Suppress auditing for coding standard xx

Where xx is any number between 1 and 37

2.2.13 Subroutine ADCON

Called by: FAUDIT

Subroutine called: ADERR

Functional Description - ADCON examines the array COL which contains an input source statement and all of its continuations, for adherence to coding standard #4 (continuation sequence numbers to be 1-9, A-J). The extracted continuation sequence numbers are compared with constant arrays DIGIT and ALPH to verify standard conformance.

2.2.14 Subroutine ADCOL

Called by: FAUDIT

Subroutines called: ADERR; ADDAT

Functional Description - ADCOL is called into execution by FAUDIT, upon encountering a "COMMON" declarative within the user's source code. Upon entry, it insures that all program "COMMON's" are labeled "COMMON's" (standard #25). Subroutine ADCOL computes the "COMMON" block's length and determines if there are other declaratives which reference "COMMON" block variables. Upon return from subroutine ADDAT, ADCOL takes one of two actions. If the "COMMON" statement is unlabeled, control is returned to the calling driver, FAUDIT. If the "COMMON" statement is labeled, the "COMMON" name and its corresponding length are checked against previously stored "COMMON" block names and lengths to insure adherence to coding standard #24 ("COMMON" block length must be the same in all modules).

2.2.15 Subroutine ADDOL

Called by: FAUDIT

Subroutine called: ADERR

Functional Description - ADDOL saves the "DO" labels of all active "DO" statements. Whenever a "DO" statement is detected by FAUDIT, ADDOL is called into execution and saves the terminating label of the "DO". The active "DO" counter, NDO, is incremented by one. Whenever the calling module, FAUDIT, encounters a labeled statement, the label value is compared with those saved by ADDOL. Upon finding a match, the active "DO" counter, NDO, is decremented. Coding standard #18 ("DO" loop nests must not exceed six levels) is violated whenever NDO exceeds six.

2.2.16 Subroutine ADRY1

Called by: ADRAY

Subroutines called: ADRY3; ADRY2

Functional Description - The statement array STATMT and "COMMON" **resident** delimiters INIT and ISTOP identify a character string to be processed by ADRY1. The character string is compared with entries in arrays IIVAR, JJVAR, LLVAR, KKVAR, and MMVAR to determine if the variable or array name presented is "INTEGER", "REAL", "LOGICAL", "DOUBLE PRECISION", or "COMPLEX" respectively. Upon determining the declared variable or array type, a corresponding numeric code is assigned (IA), the index of the appropriate array is established (IB), and subroutines ADRY3 and conditionally ADRY2 are called to further process the string.

2.2.17 Subroutine ADRY2

Called by: ADRY1

Subroutine called: ADERR

Functional Description - ADRY2 is called by ADRY1 to process a variable name if it has been determined that the variable name is an array. ADRY1's prior call to ADRY3 determines if the variable name is an array, i.e., has subscripts. ADRY2 scans for a left parenthesis following the variable name to insure adherence to coding standard #26 (never omit subscripts except in "CALL", "I/O", or "DATA" statements). Additionally ADRY2 isolates and checks each subscript element for adherence to coding standard #8 (use integer subscripts when referencing an array).

2.2.18 Subroutine ADRY3

Called by: ADRY1

Functional Description - The "COMMON" resident variable IA, set in ADRY1, declares the type of the variable to be examined by ADRY3. Correspondingly, the appropriate array (IVAR, JVAR, KVAR, LVAR, or MVAR) is examined to determine the number of subscripts (ISUB) declared for the variable.

2.2.19 Subroutine ADSTA

Called by: FAUDIT

Subroutine called: ADERR

Functional Description - ADSTA is called each time a labeled statement is encountered. The numeric value of the statement label is stored in NUMB2 and compared with NUMB1, the last legal statement label encountered. If NUMB2 is less than NUMB1, the statement is flagged as violating coding standard #3 (statement labels shall be assigned in ascending order). If NUMB2 is greater than NUMB1, then NUMB2's value is assigned to NUMB1 prior to release of execution control.

2.2.20 Subroutine ADSTR

Called by: FAUDIT

Subroutine called: GETWRD

File: IOUTP

Functional Description - ADSTR is called for each module of the user's source code processed, specifically, the "SUBROUTINE" or "FUNCTION" identifying statements. The module name is stored in array PROGNM and the module counter, NDEX, is incremented by one and tested for a value in excess of 100. An appropriate error message is printed for values greater than 100.

2.2.21 Subroutine ARGU

Called by: FAUDIT

Subroutines called: GETWRD; ADERR

Functional Description - ARGU examines the argument list of "CALL" statements to insure that the list is devoid of arithmetic or logical expressions.

2.2.22 Subroutine GETWRD

Called by: FAUDIT; ADIMP; ADIOE; ADMIX; ADSTR; ARGU

Functional Description - GETWRD accepts as input the "COMMON" resident pointer JSAVE and scans the array STATMT from that point until a delimiter is encountered. The contained character string is "ENCODED" into the variable STATE prior to return of execution control.

2.2.23 Function Subprogram NTYPE

Called by: ADMIX

Functional Description - Via "FUNCTION" formal parameters L1 and L2, **(variable name delimiters)** NTYPE determines the variable type for subsequent processing by subroutine ADMIX. The numeric values assigned per variable type are as follows:

VARIABLE TYPE	NUMERIC CODE
"INTEGER"	1
"DOUBLE PRECISION"	2
"REAL"	3
"COMPLEX"	4
"LOGICAL"	5

2.2.24 Subroutine EXPON

Called by: FAUDIT

Subroutine called: ADERR

Functional Description - EXPON's formal parameter, IX, points to an exponent to be examined. EXPON scans the array STATMT in search of a hollerith period, indicating non-conformance with coding standard #15 (whole numbers used as exponents must be integer). The scan is terminated upon encountering a special character and control is returned to the calling program, FAUDIT.

2.2.25 Subroutine TYPES

Called by: FAUDIT

Subroutine called: ADERR

File: IOUTP

Functional Description - TYPES classifies the variables of assignment expressions according to the contents of various arrays initialized during declaration statement processing.

2.2.26 Subroutine ADERR

Called by: FAUDIT; ADDAT; ADGOT; ADIOE; ADMIX; ADCON; ADCOL; ADDOL; ADRY2; ADSTA; ARGU; EXPON; TYPES

File: IOUTP

Functional Description - ADERR accepts the input variable NUMERR, an integer identifying a coding standard which has been violated. ADERR prints the error identification and updates the appropriate error counters.

2.2.27 Subroutine ADINP

Called by: FAUDIT

Files: NTAPE, IOUTP

Functional Description - ADINP is the input routine for the user's source code. It maintains two "COMMON" resident storage buffers, COL and STATMT, and associated pointers, ISTART, IEND, and INXX. COL is initially loaded with twenty user source card images. It is sized such that it may contain a FORTRAN source statement and all of its continuation cards (19 maximum). The pointers ISTART and IEND delimit a source statement and its continuations for further processing by Code Auditor modules. As each source statement is processed, ISTART and IEND are adjusted by ADINP to delimit the next source statement to be processed. In the event that IEND points to the twentieth entry of COL and that entry is a continuation card, the buffer COL is reallocated by moving the delimited statement to position 1 of COL and additional cards are read into the buffer. IEND is then re-computed to provide the upper pointer for the source statement. The buffer STATMT contains the source statement delimited by ISTART and IEND exclusive of its label, if any. All blanks are removed and the pointer INXX is set to the last character of the source statement within the single-dimensioned buffer.

2.2.28 Subroutine ADTRM

Called by: FAUDIT

File: IOUTP

Functional Description - ADTRM is called at the conclusion of Overlay A processing and prints summary reports for that processing phase of Code Auditor which examines the user's code for adherence to program coding standards.

2.2.29 Subroutine AUDIT

Called by: MAINLD

Subroutines called: INITAL (MI3PQ, IALLFG, LASTSB, LENGTH, VSTR); REINIT (IBDATA, LSTSEG, VSTR, LENGTH, ITSTX, IDONE); TABGEN (VSTR); PCARD1 (NUMCOM, IDONE, ISW); RCARDN (IDONE, NUMCOM, ISW); PCARDN (ILAST, ITHIS, ITSTX, IBDATA, MI3PQ); TRAPIT (VSTR, MIRPLS); NINSTR (c); ENDPRC (VSTR); TERMIN (VSTR)

File: OLDSRC

Functional Description - AUDIT either directly or indirectly controls and sequences the execution of all other subprograms of Overlay B of the Code Auditor.

2.2.30 Subroutine ADJUST

Called by: STLBL; ENDPRC

Parameters: I, VSTR

File: 06

Functional Description - ADJUST dynamically reallocates VSTR storage assignments for sub-tables SEGTAB, SEGBT, and SEGBF.

2.2.31 Subroutine ASSIGN

Called by: REINIT

Parameters: VSTR, LENGTH

File: 06

Functional description - Code Auditor uses the single-dimensioned array VSTR to dynamically allocate storage space for the three sub-tables, SEGTAB, SEGBF, SEGBT. A pass-one inspection of a user's source module by the Code Auditor results in a determination of the number of logical segments within the module. This figure is used by ASSIGN to allocate space for each of the three sub-tables. Sub-table SEGBF may have its space reallocated since SEGBT receives its entries after SEGBF is completed.

2.2.32 Block Data Subprograms BLK1, BLK2, BLK3, BLK4, and BLKCHR

Functional Description - Data is entered into Overlay B labeled "COMMON" areas prior to Overlay B execution by the use of "BLOCK DATA" subprograms BLK1, BLK2, BLK3, BLK4, and BLKCHR.

2.2.33 Subroutine BUILD

Called by: STNTBL

Parameters: NAME, NEWNAM

Functional Description - BUILD packs the 8 position character array NAME, into the single variable NEWNAM, left justified, blank-filled.

2.2.34 Subroutine CLENUP

Called by: ENDPRC

Parameters: SEGTAB, SEGBT, SEGBF, IDIFF

File: KPR

Functional Description - CLENUP processes VSTR's sub-table SEGBT, containing segment identification of segments branched-to, to generate entries for SEGBF, which contains segment identification of segments branched-from.

2.2.35 Subroutine DOPROG

Called by: TABGEN

File: KPR

Functional Description - DOPROG is called into execution for each "DO" statement encountered. The statement is scanned and its "DO" termination label is extracted and stored in the array IDOXPT. IDOXPT also contains the corresponding segment identification.

2.2.36 Subroutine SBXPRG

Called by: TABGEN

Parameters: LINE, SEGTAB, SEGBT, BETAB, NEWEND

Subroutines called: ADJUST (c); IFCK (LINE, I); STLBL (LINE, K, SEGBT, SEGTAB, NEWEND)

File: 06

Functional Description - SBXPRG processes the segment's 'exit' statement and stores the branch type and branch-to entries of array SEGTAB.

2.2.37 Subroutine STLBL

Called by: SBXPRG;

Parameters: LINE, J, SEGBT, SEFTAB, NEWEND

Subroutines called: ADJUST (c); BCDINT (LINE, J, KWORD, N)

Functional Description - Branch statements such as "GOTO's" and computed "GOTO's" have their labels stored in the array SEGBT by STLBL. The label value, converted from hollerith to integer by the subroutine BCDINT, is stored as a negative value for subsequent replacement with segment identification codes by subroutine CLENUP.

2.2.38 Subroutine TABGEN

Called by: AUDIT; ENDPRC; TAPWRT

Parameter: VSTR

Subroutines called: IFCK (NLINE2, J); TYPCVT (I, STYPE, NLINE2, J, NCHARS); INITSG (IFLAG, NUMBR1, ISUB, VSTR, NSGTAB); REREAD (NLINE2, NLINE1, NUMBER, NUMBR1, NUMBR2, ITYP, IFTYP, IFLAG); DOPROG; SRXPRG (NLINE2, INDEX, VSTR, NSGTAB, VSTR, NSEGBT, MBETAB, NEWEND)

File: 06

Functional Description - TABGEN provides the driver function for the generation of SEGTAB, the Program's Anatomy Table.

2.2.39 Subroutine TAPWRT, TPEOF (entry)

Called by: ENDPRC

Parameters: SEGTAB, VSTR

File: KSEGBT

Functional Description - TAPWRT writes the Segment Transfer Table to a temporary disk file for subsequent processing by Overlay C.

2.2.40 Subroutine UNBILD

Called by: KT2ND

Parameters: NAME, NEWNAM

Functional Description - UNBILD unpacks the formal parameter NAME into the formal array NEWNAM.

2.2.41 Subroutine TYPCVT

Called by: TABGEN

Parameters: ITYP, STYPE, IBUFF, IEND

File: 06

Functional Description - TYPCVT converts numeric type assignments from those required by the first pass to those required by the second pass.

2.2.42 Subroutine INITSG

Called by: TABGEN

Parameters: IFLAG, NUMBR1, NAME, SEGTAB

File: KPR

Functional Description - INITSG performs initialization procedures for each segment of a program module. Parameters' descriptive of each segment are stored in the sub-table SEGTAB.

2.2.43 Subroutine REREAD

Called by: TABGEN; PCARD1

Parameters: NLINE2, NLINE1, NUMBER, NUMBR1, NUMBR2, ITYP, IFTYP, IFLAG, STYPE, IOPT

Functional Description - REREAD processes source statements as follows depending upon the formal switch parameter IOPT. If IOPT = 0, the current source statement, in packed and unpacked format, is stored along with various characteristics of the statement. If IOPT = 1, REREAD additionally stores the previous source statement and its characteristics.

2.2.44 Subroutine POINTR

Called by: PASS1; REINIT

Parameters: IPACK, IMASK, IOPT, N1, N2

File: 06

Functional Description - Depending upon the formal parameter IOPT, POINTR uses the mask, MASK, to pack the formal parameters N1 and N2 into the single-dimensioned variable IPACK (IOPT=2), or unpack from IPACK to N1 and N2 (IOPT≠2).

2.2.45 Subroutine NINSTR

Called by: AUDIT; IFTRAP; PASS1; TRAPIT; DOTERM; PCARDN

Parameter: IOPT

Subroutine called: WRND

Functional Description - NINSTR maintains and assigns segment identification codes for user source code.

2.2.46 Subroutine IFTRAP

Called by: TRAPIT

Parameter: IFLAG

Subroutines called: IFCK (CARE, ISOS); TYPE (CARE, ISOS, JTYPE, IFTYP, c); NINSTR (c); WRND

Functional Description - IFTRAP processes "IF" statements and makes appropriate subroutine calls for 'consequent' statement type determination and segment identification code assignment.

2.2.47 Subroutine PASS1

Called by: INITAL

Parameters: VSTR, IALLFG, ISUBKNT

Subroutines called: BCDINT (TEMP, NOS, I); SQZB (CARD, c, IEND, CARE, ISQEND); TYPE (CARE, JTYPE, ITYP, ITSTX); NINSTR (c); TRAPIT (VSTR, MIRPLC); KTGEN (c, NODE, JNOD)

Files: OLDSRC, KPR

Functional Description - Via calls to subroutines SQZB, TYPE, NINSTR, TRAPIT and KTGEN, PASS1 performs the initial pass over the user's source code. It inputs each source statement, removes all blanks (SQZB), determines the statement type (TYPE), identifies segment 'entry' statements and assigns a segment number accordingly (NINSTR and TRAPIT).

2.2.48 Subroutine STNTBL

Called by: TRAPIT; PCARDN

Subroutines called: BUILD (NAM, KKTAB, JNOD); KTGEN (c, NODE, JNOD); KT2ND (NAM, LNODE, NNODE, JNOD, c)

Functional Description - STNTBL is called for every new subprogram module encountered. It uses the utility subroutine BUILD to store the module name in the array KTAB (KKTAB equivalenced), in addition to the ending segment number for the previous module and the beginning segment number for the current module (via KTGEN and KT2ND).

2.2.49 Subroutine TRAPIT

Called by: AUDIT; PASS1

Parameter: IREPLC

Subroutines called: NINSTR (c); STNTBL; WRND; IFTRAP (IFLAG)

File: 06

Functional Description - Depending upon statement classification (ITYP), TRAPIT branches accordingly performing the following function:

(1) For "SUBROUTINE" or "FUNCTION" statements, the module is stored via call to STNTBL.

(2) For all statements identified as segment 'entry' statements, node (segment) numbers are assigned

(3) "DO" statements are processed.

2.2.50 Subroutine WRND

Called by: NINSTR; IFTRAP; TRAPIT

Functional Description - WRND increments assigned node (segment 'entry' statement) numbers upon each call.

2.2.51 Subroutine KTGEN

Called by: PASS1; STNTBL

Parameters: LNODE, NNODE, JNOD

Functional Description - For each module processed during pass one, KTGEN stores the module name and delimiting segment numbers in the array KTAB.

2.2.52 Subroutine KT2ND

Called by: STNTBL

Parameters: NAM, LNODE, NNODE, JNOD, IOPT

Subroutine called: UNBILD (KKTAB, JNOD, NAM)

Functional Description - Via a subroutine call to UNBILD, KT2ND extracts from the array KTAB (KKTAB equivalenced) the name (NAM) of the module which has just undergone pass one processing. In addition, the delimiting segment numbers for the module are stored in formal paramters NNODE and LNODE respectively.

2.2.53 Subroutine DOTERM

Called by: TRAPIT

Parameters: PROC, IREPLC, J

Subroutine called: NINSTR (c)

Functional Description - DOTERM matches labels of "DO" terminators with values saved in the array IREPLC. IREPLC contains "DO" terminator values established when "DO" statements are processed.

2.2.54 Subroutine INITAL

Called by: AUDIT

Parameters: MI3PQ, IALLFG, LASTSB, LENGTH, VSTR

Subroutine called: PASS1 (VSTR, IALLFG, ISUBKNT)

File: OLDSRC

Functional Description - Via call to subroutine PASS1 and internal assignment expressions, INITAL provides system initialization and the initial pass over the user's source code.

2.2.55 Subroutine PCARD1

Called by: AUDIT

Parameters: NUMCOM, IDONE, ISW

Subroutines called: REREAD (CARE, CARD, NOS, NUMBR1, NUMBR2, ITYP, IFTYP, IFLAG, STYPE, c); BCDINT (TEMP, NOS, I)

File: KPR

Functional Description - PCARD1 prints the current statement header card image and where required, any node (segment 'entry' statement) numbers. The printer header is subsequently transferred from array TEMP to CARD for subsequent packing. All labeled statements have their labels converted to integer and stored in the "COMMON" resident variable, NOS.

2.2.56 Subroutine PCARDN

Called by: AUDIT

Parameters: ILAST, ITHIS, ITSTX, IBDATA, MI3PQ

Subroutines called: SQZB (CARD, c, IEND, CARE, ISQEND); TYPE (CARE, JTYPE, ITYP, ITSTX); NINSTR (c); STMTBL

Functional Description - PCARDN packs the current user's source statement via a subroutine call to SQZB. The input statement image, unpacked, resides in the array CARD prior to the SQZB call, delimited by the argument IEND. Upon return of execution control from SQZB, the current statement image resides in the array CARE, packed, delimited by the argument ISQEND. A subsequent call to subroutine TYPE returns numeric codes indicative of statement type (JTYPE), classification (ITYP), and whether the statement is the first executable statement of the source module (ITSTX=2). If the current statement is within a "BLOCK DATA" subprogram, it is ignored. Likewise, an "END" statement is ignored. If the statement is labeled it is designated an 'entry' statement and assigned the next sequential node number. If the statement is the initial statement of a module, i.e., "FUNCTION", "SUBROUTINE", etc., it is ignored.

2.2.57 Subroutine REINIT

Called by: AUDIT

Parameters: IBDATA, LSTSEG, VSTR, LENGTH, ITSTX, IDONE

Subroutines called: POINTR (VSTR, I, c, I1, I2); ASSIGN (VSTR, LENGTH)

File: KPR

Functional Description - REINIT performs the necessary variable reinitialization for each new source module processed.

2.2.58 Subroutine TERMIN

Called by: AUDIT

Parameter: VSTR

Subroutine called: TAPWRT (TPEOF)

Functional Description - TERMIN is called at the conclusion of Overlay B processing to write an end-of-file record on the temporary mass storage device containing Segment Transfer Table.

2.2.59 Subroutine RCARDN

Called by: AUDIT

Parameters: IDONE, NUMCOM, ISW

File: OLDSRC

Functional Description - RCARDN reads continuation cards of the current statement being processed by Overlay B of the Code Auditor. Its processing includes a feature for ignoring command cards imbedded among the current statement's continuation cards.

2.2.60 Subroutine ENDPRC

Called by: AUDIT

Parameter: VSTR

Subroutines called: TABGEN (VSTR); ADJUST (IDIFF, VSTR); CLENUP (VSTR, MSGTAB, MSEGBT, MSEGBF, IDIFF); TAPWRT (VSTR, MSGTAB)

File: 06

Functional Description - ENDPRC is called into execution whenever a FORTRAN "END" card is encountered. It tests the storage array VSTR for the last entry of sub-table SEGBT, and establishes the next available storage location as the origin for sub-table SEGBF, i.e., SEGBF's space is dynamically reallocated. Subsequent calls are made to subroutine CLENUP, to complete VSTR table entry, and TAPWRT, to write selective VSTR table entry, and TAPWRT, to write selective VSTR data to a temporary mass storage file for Overlay C input.

2.2.61 Subroutine BCDINT

Called by: PASS1; PCARD1; STLBL; TABGEN

Parameters: IBUFF, INTEG, NEND

Functional Description - BCDINT accepts as input the formal parameters IBUFF and NEND. Its function is to examine the first NEND characters of the array IBUFF and convert the hollerith string of numbers to its integer value. If the hollerith string contains a non-digit first character, the parameter NEND has its value set to the negative of the character's position in IBUFF, and control is returned. Otherwise, the numeric value of the character string is computed and returned via the formal parameter INTEG.

2.2.62 Subroutine IFCK

Called by: SBXPRG; TABGEN; IFTRAP

Parameters: IBUFF, LASTRP

File: 06

Functional Description - IFCK accepts as input the array IBUFF passed as a formal parameter. IBUFF contains the current user source statement being processed. IFCK is called into execution by various Code Auditor routines to process logical "IF" expressions. IFCK scans the array in search of left and right parentheses, terminating the scan whenever the number of left parentheses encountered match the number of right parentheses. The numeric position in IBUFF of the appropriate right parenthesis is returned via the formal parameter LASTRP.

2.2.63 Subroutine SQZB

Called by: PASS1; PCARDN

Parameters: I1BUFF, I1BEG, I1END, I2BUFF, I2END

File: 06

Functional Description - SQZB's input formal parameters, I1BUFF, I1BEG, and I1END describe a buffer and delimiters respectively in passing a character string to be examined. The character string contained by I1BUFF beginning at location I1BEG is transferred to the buffer I2BUFF.

Blank characters are not transferred. Transfer of characters terminates after I1END characters are transferred. I2BUFF's entries begin in cell 1 and terminates in cell I2END.

2.2.64 Subroutine TYPE

Called by: PASS1; PCARDN

Parameters: IBUFF, JTYPE, ITYP, ITSTX

Subroutine called: IFCK (IBUFF, LASTRP)

Functional Description - TYPE accepts as input the formal argument, IBUFF, containing a packed source statement (i.e., no intervening blanks) for type determination. The first alphabetic character of the input character string in IBUFF is used to generate an index, subsequently used in a computed "GOTO". The string is then analyzed for the presence of a FORTRAN keyword. Upon finding the appropriate keyword, the formal parameters, JTYPE and ITYP are set in accordance with the following tables. JTYPE is an integer specifying the statement type while ITYP specifies the statement classification.

JTYPE	ITYP	KEYWORD
1	0	"ASSIGN"
2	0	"BACKSPACE"
3	16	"BLOCK DATA"
4	9	"CALL"
5	8	"CALL EXIT"
6	0	"COMMON"
7	0	"COMPLEX"
8	13	"CONTINUE"
9	0	"DATA"
10	0	"DECODE"
11	0	"DIMENSION"

JTYPE	ITYP	KEYWORD
12	4	"DO"
13	0	"DOUBLE PRECISION"
14	0	"ENCODE"
15	3	"END"
16	0	"ENDFILE"
17	15	"ENTRY"
18	0	"EQUIVALENCE"
19	0	"EXTERNAL"
20	2	"FUNCTION"
21	14	"FORMAT"
22	7	"GO TO"
23	6	"GO TO" (computed)
24	5	"IF"
25	1	"IMPLICIT"
26	0	"INTEGER"
27	-	NOT USED
28	0	"LOGICAL"
29	0	"NAMELIST"
30	12	"PRINT"
31	0	"PARAMETER"
32	0	"PUNCH"
33	12	"READ"
34	0	"REAL"
35	11	"RETURN"
36	0	"REWIND"

JTYPE	ITYP	KEYWORD
37	10	"STOP"
38	2	"SUBROUTINE"
39	12	"WRITE"

2.2.65 Subroutine STRUCT

Called by: MAINLD

Subroutine called: LPEX

Files: KSEGBT, KPR

Functional Description - STRUCT accepts as input the Segment Transfer Table generated in Overlay B and applies the reduction algorithm to make a determination of the corresponding module's conformance to 'structured program' requirements

2.2.66 Subroutine LPEX

Called by: STRUCT

Functional Description - LPEX is a sub-function of STRUCT and it verifies a loop as having more than one exit.

2.3 Major System Variables

The Major System Variables are described in two groups, labeled "COMMON" (2.3.1) and blank "COMMON" (2.3.2).

Four labeled "COMMON's" (AUDCOM, BK0134, BKFOUR, and BKZERO) and their associated variables are further defined in Tables II through V to indicate their relationships to "EQUIVALENCED" variables. These "EQUIVALENCED" variables bear the functional references by the various modules in which these labeled "COMMON's" occur.

For example: Main Program - COMMON/X/A(5)

```
EQUIVALENCE (A(1), B), (A(2) C), (A(3) D)
CALL SUB1
STOP
END
SUBROUTINE SUB1
COMMON/X/A(5)
EQUIVALENCE (A(1), R), (A(4), S)
R = 5.0
S = R*6.0
RETURN
END
```

The labeled "COMMON"name X provides for storage of the associated array variable A. The associated array variable A is then "EQUIVALENCED" to individual variables for subsequent reference in module SUB1.

This scheme for "COMMON" definition is used for labeled "COMMON's" AUDCOM, BK1034, BKFOUR, and BKZERO only. The associated array variables are AUDCOM, QACOM5, QACOM4, and QACOM0, respectively.

The blank "COMMON" variables IVAR, JTAB, JVAR, KTAB, KVAR, LVAR, and MVAR are "EQUIVALENCED" to structure variables. These "EQUIVALENCED" structure variables are superimposed over the subject variables to facilitate storage reference of two words. This allows capturing routine names and variable names which may consist of a maximum of eight characters. Eight characters exceed the character storage capacity of one storage word thereby requiring "CHARACTER" typing of the "EQUIVALENCED" structure variables to twelve enabling reference of two storage words. These variables are indicated under the "EQUIVALENCED" structure variable column of the blank "COMMON" (2.3.2) paragraph.

Some of the RADC Code Auditor variables were inactivated due to conversion. These variables were retained in the code to allow easy update of functions stripped during conversion, assuming the potential requirement to incorporate the functions at a later date. These variables are indicated as inactive.

2.3.1 Labeled "Commons"

Label	Occurs in Routine	Associated Variable(s)	Description
A0	FAUDIT; ADEND; ARGU; TYPES	A0	Total dimension of JTAB table.
		A1	Pointer to first dimension of JTAB variable.
		A2	Pointer to second dimension of JTAB variable.
		A3	Pointer to third dimension of JTAB variable.
		A4	Pointer to type of JTAB variable.
ALPH	FAUDIT; ADIMP; ADIOE; ADMIX; ADRAY; ADCCO; ADCON; ADRY2; NTYPE; TYPES; ADINP	ALPH	Hollerith array containing the 26 alphabets.
AUDCOM	See Table II		
B0	ADEND; ADRY3; NTYPE; TYPES	B0	Total dimension of the "INTEGER" (IVAR) and "REAL". (JVAR) variable typing arrays.
		B1	Pointer to first dimension of "INTEGER" (IVAR) and "REAL" (JVAR) variable typing arrays.
		B2	Pointer to second dimension of "INTEGER" (IVAR) and "REAL" (JVAR) variable typing arrays.
		B3	Pointer to third dimension of "INTEGER" (IVAR) and "REAL" (JVAR) variable typing arrays.
BK0134	See Table III		
BKFOUR	See Table IV		
BKTEMP	AUDIT; DOPROG; SBXPRG	IDOXPT	"DO" terminator storage array.
		LDO	Number of entries in IDOXPT array.
BKZERO	See Table V		
BLOCKN	ADCOL	BLOCKN	Array containing "COMMON" block names.

2.3.1 Labeled "Commons" (Continued)

Label	Occurs in Routine	Associated Variable(s)	Description
CO	ADEND; ADRY3; NTYPE; TYPES	CO	Total dimension of the "DOUBLE PRECISION" (KVAR), "LOGICAL" (LVAR) and "COMPLEX" (MVAR) variable typing arrays.
		C1	Pointer to first dimension of "DOUBLE PRECISION" (KVAR), "LOGICAL" (LVAR) and "COMPLEX" (MVAR) variable typing arrays.
		C2	Pointer to second dimension of "DOUBLE PRECISION" (KVAR), "LOGICAL" (LVAR) and "COMPLEX" (MVAR) variable typing arrays.
		C3	Pointer to third dimension of "DOUBLE PRECISION" (KVAR), "LOGICAL" (LVAR) and "COMPLEX" (MVAR) variable typing arrays.
COL	FAUDIT; ADCCD; ADCON; ADERR; ADINP	COL	Card image buffer for audited source, twenty cards read in per read unless continuation cards are at the end of the buffer. ADINP responsible for buffer control.
DELIM	FAUDIT; ADMIX; ADRAY; ARGU; GETWRD; EXPON	DELIM	Array containing FORTRAN delimeters (12 hollerith characters).
DIGIT	FAUDIT; ADIOE; ADMIX; ADRAY; ADCCD; ADCOL; ADRY2; NTYPE	DIGIT	Array containing 10 hollerith digits.
DOLAB	FAUDIT; ADENP	DOLAB	Array of active "DO" statement labels.
FREQ	IFTRAP	FREQ	Inactive.
FRTF	FAUDIT	FRTF	Array containing FORTRAN subroutines not included in mathematical library routines, i.e., TIME, RANGET, etc.
FUNC	NTYPE	FUNC	Array containing "INTEGER" FORTRAN library subroutine names.

2.3.1 Labeled "Commons" (Continued)

Label	Occurs in Routine	Associated Variable(s)	Description
FUNCC	NTYPE	FUNCC	Array containing "COMPLEX" FORTRAN library subroutine names.
FUNC	ADMIX	FUNC	Array containing "LOGICAL" FORTRAN library subroutine names.
FUNCM	NTYPE	FUNCM	Array containing "DOUBLE PRECISION" FORTRAN library subroutine names.
FUNCR	NTYPE	FUNCR	Array containing "REAL" FORTRAN library subroutine names.
FUNNAM	FAUDIT; ADEND; NTYPE	FUNNAM	Storage for module statement function name.
I	ADRAY; ADRY1; ADRY2	I	"COMMON" index used in subroutines ADRAY, ADRY1, ADRY2.
I11	ADIOE; GETWRD	I11	Pointer to first character of word in STATMT array.
IA	ADRY1; ADRY2; ADRY3	IA	Pointer indicating located variable typing array (IVAR or JVAR or KVAR or LVAR or MVAR).
IB	ADRY1; ADRY2; ADRY3	IB	Pointer indicating entry of located variable typing array (IVAR or JVAR or KVAR or LVAR or MVAR).
ICALLN	IFTRAP; TRAPIT	ICALLN	Inactive.
ICOMN	FAUDIT; ADEND	ICOMN	Flag used to indicate occurrence of preface commentary analysis.

2.3.1 Labeled "Commons" (Continued)

Label	Occurs in Routine	Associated Variable(s)	Description
IDBLST	ADINP	IDBLST	Flag indicating 2 statements separated by $ delimeter. = 0, only 1 statement per card. = 1, two or more statements per card.
IDEBUG	FAUDIT; ADEND; ADIMP; ADRAY; ADCCD; ADCOL; ADRY1; ADRY2; ADSTA; ADSTR; ADINP	IDEBUG	Flag which indicates program execution in debug mode causing messages to be printed containing values of various key variables.
IDIM	ADCOL	IDIM	Used to save accumulated dimensions of variables.
IEIGHTY	ADCCD	IEIGHTY	Contains the constant value of 80 for column length of card.
IEND	FAUDIT; ADCCD; ADCON; ADERR; ADINP	IEND	Position of last continuation card in array COL.
IENDQ	FAUDIT; ADIMP; ADIOE; ADMIX; ADSTR; ARGU; GETWRD	IENDQ	Pointer to last character of word contained in STATE.
IEOF	FAUDIT; ADCCD; ADINP	IEOF	Indicator for end-of-file.
IEQU	FAUDIT	IEQU	Counter for "=" signs in assignment statements.
IEXA	ADEND	IEXA	Total number of errors detected for the number of routines processed.
IEXT	FAUDIT; ADEND; ADINP	IEXT	Total number of executable statements per module.
IFCOUN	FAUDIT; ADEND	IFCOUN	Total number of "IF" statements per module.

2.3.1 Labeled "Commons" (Continued)

Label	Occurs in Routine	Associated Variable(s)	Description
IFORMT	FAUDIT; ADEND	IFORMT	"FORMAT" statement indicator.
IFRCC	FAUDIT; ADEND	IFRCC	= 1, incorrect first card of preface commentary block.
IHOLT	FAUDIT; ADMIX; ADRAY; ADINP	IHOLT	Indicator for embedded hollerith character = 0, not hollerith character. = 1, hollerith character.
IKOM	FAUDIT; ADEND	IKOM	Counter for first comment card after "PROGRAM", "SUBROUTINE" or "FUNCTION" card. Code Auditor allows at least two non-comment cards.
IKOUNT	FAUDIT; ADEND; ADCCD; ADERR; ADINP	IKOUNT	Card count per module.
ILEVEL	FAUDIT; ADEND	ILEVEL	Originally used to indicate commentary level; currently inactive.
ILSTCD	ADCCD; ADINP	ILSTCD	Pointer to last card in input buffer before end-of-file.
IMP1	ADEND; ADIMP; ADIOE; NTYPE; TYPES	IMP1	"IMPLICIT" typing array containing an entry for each alphabetic character with "IMPLICIT" statement; if IMP1(I) = ITYPE, variables starting with the IMP1(I) letter are implicitly typed.
IMPYES	FAUDIT; ADEND; ADIOE; ADMIX; NTYPE	IMPYES	"IMPLICIT" indicator. = 0, no "IMPLICIT" card. = 1, "IMPLICIT" card encountered.

2.3.1 Labeled "Commons" (Continued)

Label	Occurs in Routine	Associated Variable(s)	Description
INAME	ADRAY; ADRY2	INAME	= 0, array name is being processed. = 1, subscripts are being processed.
IND	PASS1; REINIT	IND	Inactive.
INIT	ADRAY; ADRY1, ADRY2	INIT	Pointer to first character of possible array name or hollerith string.
INPUT	ADCCD; INITAL	INPUT	Logical input file (set = to 5).
INXX	FAUDIT; ADEND; ADGOT; ADIMP ADMIX; ADRAY; ADCOL; ADRY2 ARGU; GETWRD; EXPON; ADINP	INXX	Last column of statement in STATMT array.
INYY	FAUDIT; ADINP	INYY	First column of statement in STATMT array (if greater than 7, reset to 7).
IOPCK	MAINLO; FAUDIT; ADCCD	NCA	Option control for standards processing. = 0, audit for stadards verification. = 1, do not audit for standards verification.
		NPA	Option control for structural processing. = 0, audit for structural processing. = 1, do not audit for structural processing.
IOUTP	FAUDIT; ADEND; ADIMP; ADIOE; ADRAY; ADCCD; ANCOL; ADRY1; ADRY2; ADSTA; ADSTR; ADERR; ADINP; ADTRM	IOUTP	Logical output file (set = to 6).
IPCL	FAUDIT; ADCCD; ADRY2	IPCL	Inactive.
IPROG	TRAPIT	IPROG	Inactive.

2.3.1 Labeled "Commons" (Continued)

Label	Occurs in Routine	Associated Variable(s)	Description
IREAL	FAUDIT; ADGOT; ADCCD; ADCOL; ADRY2	IREAL	Indicator for real time users. = 1, real time user. = 0, non-real time user (set in ADCCD).
ISAVE	ADCOL	ISAVE	Used to preserve the value of JSAVE.
ISLASH	ADCOL	ISLASH	Counter for slashes.
ISTART	FAUDIT; ADCON; ADERR; ADINP	ISTART	Identifies the initial structure of the array COL, a buffer which contains 20 FORTRAN card images.
ISTATE	FAUDIT; ARGU; GETWRD; ADIMP; ADIOE	ISTATE	Encoded storage for word fetched by GETWRD for use in checking against integer variables.
ISTOP	ADRAY; ADRY1; ADRY2	ISTOP	Pointer to last character of array name.
ISUB	BLK4; TRAPIT; INITAL; ADRY1 ADRY2; ADRY3	ISUB	Number of subscripts determined for this array name.
ISUBST	FAUDIT; ADINP	ISUBST	Multiple statement indicator. = 1, more than one statement per card. = 0, one statement per card.
ITAB	FAUDIT; ADEND; ADIOE; ARGU; TYPES	ITAB	Number of JTAB array entries.
ITEST	FAUDIT; ADMIX; ADRAY; ADINP	ITEST	Index to character BEINT tested to see if it is embedded in hollerith; ITEST must be zero for a call which is not making a hollerith check.

2.3.1 Labeled "Commons" (Continued)

Label	Occurs in Routine	Associated Variable(s)	Description
ITFLG	FAUDIT; ADEND	ITFLG	Indicator used to call module types once per module.
ITOTL	ADEND; ADERR	ITOTL	Contains total number of errors per standard per routine.
ITRACE	TRAPIT	ITRACE	Inactive.
ITYPE	FAUDIT; ADCOL	ITYPE	Variable type indicator. = 1, "INTEGER". = 2, "DOUBLE PRECISION". = 3, "REAL". = 4, "COMPLEX". = 5, "LOGICAL".
JCONNT	IFTRAP; PASS1	JCOUNT	Array containing count of "statement types" encountered.
JJ	FANOIT; ADMIX	JJ	Pointer to right parenthesis of "IF" expression identified in STATMT array.
JLST	ADEND; ADCCD; ADERR; ADINP	JLST	Indicator for listing module being processed.
JSAVE	FAUDIT; ADGOT; ADIMP; ADIOE; ADMIX; ADRAY; ADCOL; ADRY1; ARGU, GETWRD; NTYPE	JSAVE	Pointer set to column following word being processed (set in ADIMP from GETWRD).
JTYPE	FAUDIT; ADMIX	JTYPE	Variable containing numeric value indicative of a statement's type, e.g., "COMMON", "DIMENSION", "GO TO", etc.
KANUMB	FAUDIT; ADERR; ADINP; ADTRM	KANUMB	Total card count for a module.

2.3.1 Labeled "Commons" (Continued)

Label	Occurs in Routine	Associated Variable(s)	Description
KETIND	FAUDIT; ADEND	KETIND	"RETURN" indicator containing the executable count of the "RETURN" cards per module.
KNTERR	ADEND; ADERR	KNTERR	Running count of errors per module.
KOMENT	FAUDIT; ADEND	KOMENT	Inactive.
KORIND	FAUDIT; ADEND	KORIND	Indicator for "FORMAT" cards; contains the card number of the first "FORMAT" card.
KOUNT	ADEND; ADTRM	KOUNT(J)	Total number of cards encountered for the Jth module.
KOUNTR	TRAPIT	KOUNTR	Inactive.
KOUT	ADEND, ADCCD, ADINP	KOUT	Line counter used to control page eject.
KRDCNT	ADEND	KRDCNT	Contains the number of executable statements per module.
KRDSAV	ADMIX	KRDSAV	Pointer to last position in STATMT array for an expression identified as a logical "IF".
KTYPE	FAUDIT; ADMIX	KTYPE	Integer variable depicting the type of a variable ("INTEGER", "REAL", etc.).
LASTCC	FAUDIT	LASTCC	When = 0, indicates end of commentary block.
LINESP	ADERR; ADINP	LINESP	Starting column of card for error print out (set in ADCCD from CAOPTION card).
LODIM	ADCOL	LODIM	Array containing size of all "COMMON" blocks in module.
LTYPE	FAUDIT; ADMIX	LTYPE	Variable containing numeric value required for indicating a variable's type.

2.3.1 Labeled "Commons" (Continued)

Label	Occurs in Routine	Associated Variable(s)	Description
MECAP	ADERR, ADTRM	MECAP	Inactive.
MLINSP	ADERR, ADINP	MLINSP	Inactive.
MLST	ADCCD	MLST	Inactive.
MNINETY	ADCCD	MNINETY	Remain constant and equals 90.
MNOCOA	FAUDIT; ADCCD	MNOCOA	Inactive.
MNRSUP	ADCCD	MNRSNP	Inactive.
MXERR	FAUDIT; ADMIX	MXERR	When = 1, indicates the current expression is mixed mode. = 0, indicates the current expression is not mixed mode.
NAMEFG	ADRAY; ADRY1; ADRY2	NAMEFG	= 1, possible array name. = 0, possible hollerith string. = 3, hollerith string.
NAMEP	IFTRAP; TRAPIT	NAMEP	Inactive.
NAMER	IFTRAP; TRAPIT	NAMER	Inactive.
NDEXS	FAUDIT; ADEND; ADSTR; ADERR ADTRM	NDEXS	Index for indicating each module processed.
NDO	FAUDIT; ADEND	NDO	Index for active "DO" loops.
NENT	ADEND; ADIOE; ADRY1; ADRY2 NTYPE; TYPES	NENT	Number of "INTEGER" variable names.
NENTC	ADEND; ADRY1; NTYPE; TYPES	NENTC	Number of "COMPLEX" variable names.

2.3.1 Labeled "Commons" (Continued)

Label	Occurs in Routine	Associated Variable(s)	Description
NENTDP	ADEND; ADMIX; ADRY1; NTYPE; TYPES	NENTDP	Number of "DOUBLE PRECISION" variable names.
NENTL	ADEND; ADRY1; NTYPE; TYPES	NENTL	Number of "LOGICAL" variable names.
NEWTR	ADEND; ADIOE; ADRY1; ADRY2; NTYPE; TYPES	NENTR	Net number of errors for each standard.
NERROR	ADEND; ADTRM	NERROR	Total number of standards to be audited.
NINETY	ADCCD; ADERR; ADINP	NFNETY	= 80 default (set in subroutine ADCCD).
NBLOC	ADCOL; TYPES	NOBLOC	Number of "COMMON" blocks in BLOCN.
NOCHWD	ADRY1; GETWRD	NOCHWD	Maximum number of characters in variable "CHARACTER" typed; set to 10.
NOS	FAUDIT; ADSTA	NOS	Number of characters in statement label.
NOSTOR	ADEND; ADERR	NOSTOR	Contains the Code Auditor card number where errors occurred (position in table corresponds to standard number).
NOSUB	ADRY1; ADRY2	NOSUB	Number of subscripts found in processing array name.
NPROG	BUILD; BLK4; TRAPIT	NPROG	Error summary information, first word contains name of module; words 2-20 contain total number of errors/standards.
NRES	FAUDIT; ADEND; ADGOT; ADMIX; ADCCD; ADCOL; ADRY1; ADRY2; ARGU; NTYPE; TYPES	NRES	Array representing standard numbers, (if set to one, then standard will not be processed).

2.3.1 Labeled "Commons" (Continued)

Label	Occurs in Routine	Associated Variable(s)	Description
NSUBR	TRAPIT	NSUBR	Inactive.
NSUP	FAUDIT; ADEND; ADCAN; ADERR; ADINP	NSUP	Indicator for type of code. = 0, FORTRAN code. = 1, GMAP code (Inactive).
NTAPE	MAINLD; ADCCD; ADINP	NTAPE	Logical input file for modules to be audited.
NUMB1	ADEND; ADSTA	NUMB1	First statement label.
NUMB2	ADEND; ADSTA	NUMB2	Second statement label.
NUMERR	FAUDIT; ADEND; ADGOT; ADIRE; ADMIX; ADCON; ADCXL; ADRY2; ADSTA; ARGU; EXPON; TYPES; ADERR	NUMERR	Standard number which was validated.
NXTCRD	FAUDIT; ADEND	NXTCRD	Set to 1 when "PROGRAM", "SUBROUTINE", or "FUNCTION" card is encountered to start processing of preface commentary block.
OUTPUT	INITAL	OUTPUT	Inactive.
PROGNM	FAUDIT; ADEND; ADSTR; ADTRM	PROGNM	Array containing all module names.
STAFUN	TYPE	STAFUN	Inactive.
STALAB	FAUDIT; ADSTA	STALAB	Encoded variable containing statement label.
STATE	FAUDIT; ADMIX; ADRY1; ADSTR GETWRD	STATE	Storage for word fetched and passed by GETWRD.
STATMI	ADRAY; ADRY2	STATMI	Single character of input statement being checked.

2.3.1 Labeled "Commons" (Continued)

Label	Occurs in Routine	Associated Variable(s)	Description
STATMT	FAUDIT; ADEND; ADDAT; ADIMP; ADIOE; ADMIX; ADRAY; ADCOL; ADRY1; ADRY2; ARGU; GETWRD; NTYPE; EXPON; ADIMP	STATMT	Array containing entire FORTRAN statement (dimensioned for 20 cards, 65 columns plus one [for the delimiter]).
TEXT	ADEND; ADTRM	TEXT	Array containing hollerith digits 1-47.
TRACE	IFTRAP	TRACE	Inactive.

Table II LABEL: AUDCOM

Variable	AUDCOM Offset	Occurs in Routine	Description
HEQ	141	BLK4; TRAPIT; IFTRAP	Hollerith equal sign (=).
IAST	144	BLK4; IFTRAP, PASS1, PCARD1; RCARDN	Hollerith asterisk (*).
ICDC	59	BLK4; NINSTR; STNTBL; TRAPIT; TERMIN	Inactive.
ICNT	65	BLK4; PASS1; PCARDN; TERMIN	Inactive.
ICNTF	82	BLK4; IFTRAP; PASS1; TERMIN	Inactive
ICND	1	REREAD; NINSTR; IFTRAP; PASS1; TRAPIT; PCARD1; PCARDN; IFCK	Identifies card number of current statement.
IEQU	99	BLK4; IFTRAP	= 1, equal sign (=) character encountered while scanning array CARE. = 0, equal sign (=) character not encountered while scanning array CARE.
ILGIC	62	BLK4; NINSTR; IFTRAP; INITAL	Inactive.
INDX	102	AUDII; INITAL; REINIT	Module counter used as index.
INODE	55	AUDIT; NINSTR; PASS1; WRND; INITAL; PCARD1	Segment 'entry' statement node counter.
INSERT	135	BLK4; INITAL	Inactive.
IPACK	63	BLK4; STNTBL; INITAL	Inactive.

Table II LABEL: AUDCOM (Continued)

Variable	AUDCOM Offset	Occurs in Routine	Description
IPASS	129	BLK4; NINSTR; IFTRAP; STNTBL; KTGEN; INITAL; TRAPIT	Inactive.
IPER	145	BLK4; PASS1; PCARD1; RCARDN	Hollerith period (.).
IRCNT	150	BLK4; TRAPIT; REINIT	Inactive.
ISUB	102	PASS1; BLK4	Name of module being processed.
INUNIT	130	BLK4	Inactive.
JNOD	64	BLK4; PASS1; STNTBL; TRAPIT; INITAL	KTAB index.
JNODE	56	AUDIT; NINSTR; IFTRAP; PASS1; TRAPIT; WRND; DOTERM; INITAL; PCARDN	Segment 'entry' statement node counter.
JUST	53	AUDIT; NINSTR; IFTRAP; PASS1; TRAPIT; WRND; DOTERM; INITAL; PCARDN	Indicator for node assignment.
JUSTST	147	BLK4; PASS1; DOTERM	Indicator for node assignment.
JUSTSV	146	BLK4; PASS1; TRAPIT	Indicator for node assignment.
LDT	100	AUDIT; TYPCVT; IFTRAP; BLK4; PASS1; TRAPIT	Total number of "DO" terminator labels.

Table II LABEL: AUDCOM (Continued)

Variable	AUDCOM Offset	Occurs in Routine	Description
MAIN	101	AUDIT; NINSTR; IFTRAP; PASS1; PCARDN; TRAPIT; BLK4	= 0, current module not a main program. = 1, current module is a main program.
MIRPLC	148	BLK4; PASS1; INITAL; REINIT	Number of NIRPLC elements.
MLLLL	138	BLK4; INITAL	Inactive.
MNDIN	136	BLK4; NINSTR; INITAL	Inactive.
MINDIP	135	NINSTR	Inactive.
MNDOU	137	BLK4; NINSTR; INITAL	Inactive.
MNDTP	140	BLK4; INITAL	Inactive.
MNODE	134	BLK4; NINSTR; IFTRAP; INITAL	Inactive.
MNODT	140	BLK4; NINSTR; INITAL	Inactive.
MONITR	133	BLK4; STNTBL; INITAL	Inactive.
MSUBCL	111	BLK4; INITAL	Inactive.
MTRACE	110	BLK4; INITAL	Inactive.
N	54	NINSTR; IFTRAP; PASS1; TRAPIT; INITAL; REINIT: TERMIN	Inactive.
NDIMP	57	BLK4; TRAPIT; INITAL	Inactive.
NDT	3	AUDIT; TYPCVT; IFTRAP; PASS1; TRAPIT; INITAL	Storage array for "DO" terminator labels.

Table II LABEL: AUDCOM (Continued)

Variable	AUDCOM Offset	Occurs in Routine	Description
NDOUT	58	BLK4; TRAPIT; INITAL	Inactive.
NEWSRC	142	BLK4; PASS1; INITAL; PCARD1; RCARDN	Inactive.
NIN	60	BLK4; INITAL	Inactive.
NIRPLC	149	BLK4; PASS1: TRAPIT; DOTERM; INITAL; REINIT	Counter specifying number of "DO" loop termination labels in array.
NJOB	131	BLK4	Inactive.
NNTRAP	132	BLK4; PASS1; TRAPIT	Inactive.
NODE	2	NINSTR; IFTRAP; PASS1; STNTBL; TRAPIT; WRND; DOTERM; INITAL	Node counter.
NODSUM	128	NINSTR; PASS1	Inactive.
NOTRAP	103	PASS1; TRAPIT	Inactive.
NOUT	61	BLK4	Inactive.
OLDSRC	143	AUDIT; PASS1; BLK4; INITAL; REINIT; RCARDN	Logical unit name assigned to user's source code input.

Table III LABEL: BK0134

Variable Name	QACOM5 Offset	Occurs in Routines	Description
BCD	3107	IFTRAP; PASS1; TRAPIT	The 10 decimal digits in hollerith.
BLANK	183	STNTBL	A hollerith blank.
CARD	186	NINSTR; IFTRAP; PASS1; TRAPIT; PCARDN	Array containing current statement being processed (unpacked).
CARE	1786	IFTRAP, PASS1; STNTBL; TRAPIT; PCARD1; PCARDN	Array containing current statement being processed (packed).
HO	3107	PASS1; RCARDN	Hollerith zero.
HBL	183	IFTRAP; PASS1; TRAPIT; RCARDN	Hollerith blank.
HC	156	PASS1; PCARD1; RCARDN	Hollerith character C.
HCM	184	IFTRAP; TRAPIT; DOTERM; INITAL	Hollerith comma.
HLP	180	IFTRAP; TRAPIT	Hollerith left parenthesis.
HRP	181	IFTRAP; TRAPIT	Hollerith right parenthesis.
IBLANK	183	TRAPIT	Hollerith blank parenthesis.
ICARDS	1	BLK3; REREAD	The number of cards containing the current statement.
ICLSP	181	TRAPIT	Hollerith right parenthesis.
ICRD	2	BLK3	Temporary storage location for a single input card.
IFLAG	82	AUDIT; BLK3; SBXPRG; TABGEN; PCARD1; REINIT; RCARDN	Type of "IF" statement (1 = logical, 0 = arithmetic).

Table III LABEL: BK0134 (Continued)

Variable Name	QACOM5 Offset	Occurs in Routines	Description
IFPRN1	83	BLK3; TABGEN; REREAD	Location (array position) of initial left parenthesis of an "IF" expression.
IHP	169	STNTBL	Hollerith P.
IHS	172	STNTBL	Inactive.
ILEVEL	151	BLK3; TRAPIT; INITAL	Inactive.
IROUTS	84	BLK3	Inactive.
ISEGLV	152	AUDIT; BLK3; INITAL; PCARD1; REINIT, TERMIN	Inactive.
ISQEND	185	REREAD; PASS1; PCARDN; TYPE; TABGEN; IFTRAP	Number of characters in current statement being processed
IUNITS	144	BLK3	Inactive.
KTABLE	153	BLK3; TABGEN	Inactive.
LO	3107	DOPROG	Hollerith zero.
LALPHA	154	BLK3	An array containing the alphabet (hollerith).
LBLK	183	BLK3; DOPROG; TABGEN; TAPWRT	Hollerith blank.
LBLANK	183	SBXPRG	Hollerith blank.
LCOMA	184	BLK3; SBXPRG	Hollerith comma.
LCOMMA	184	DOPROG; STLBL	Hollerith comma.
LLPRN	180	BLK3; SBXPRG	Hollerith left parenthesis.

Table III LABEL: BK0134 (Continued)

Variable Name	QACOM5 Offset	Occurs in Routines	Description
LNUMER	1308	DOPROG, SBXPRG	Hollerith decimal digits 0 through 9.
LRPRN	181	BLK3	Hollerith right parenthesis.
MAXLIN	182	BLK3	Inactive.
NCHAR	185	BLK3; DOPROG; SBXPRG; TRAPIT; PCARD1	Inactive.
NEXT	152	BLK3; TABGEN; INITAL	Hollerith character 'Next'.
NLINE1	186	BLK3; TABGEN; INITSG	Array containing current statement being processed (unpacked).
NLINE2	1786	BLK3; DOPROG; TABGEN	Inactive.
NOSUB	3106	BLK3	Number of subscripts found in processing array name.
NSTART	104	TABGEN	Inactive.
NSTOP	124	TABGEN	Inactive.
NSUB	34	TABGEN; PASS1; INITAL; REINIT	Inactive.
NUM	3107	BLK3; SBXPRG	Hollerith decimal digits 0 through 9.
NUMBIT	3117	BLK3	A constant whose value is '10'.
OPENP	180	STLBL	A constant whose value is a hollerith left parenthesis.
TEMP	2	AUDIT; PASS1; INITAL; PCARD1; REINIT; PCARDN	Temporary storage array for current statement being processed.

Table IV LABEL: BKFOUR

Variable Name	QACOM4 Offset	Occurs in Routine(s)	Description
IFFLAG	103	BLK2; SBXPRG; TABGEN; INITSG	1 = current statement is a logical "IF" statement; 0 = not a logical "IF" statement.
IFTYP	128	TABGEN; IFTRAP; PCARD1	An integer identifying the class of the statement being processed.
ITYP	127	AUDIT; TABGEN; PASS1; STNTBL; TRAPIT; PCARD1; PCARDN	An integer identifying the class of the statement being processed; e.g., declaration, branch statements, etc.
KSEGBT	104	MAINLD; TABGEN; TAPWRT; INITAL; PCARD1; PCARDN	Device name for disk which stores Segment Transfer Table for Overlay C processing.
KSFILE	105	TABGEN, TAPWRT, INITAL	Inactive.
MASK	106	BLK2; CLENUP; SBXPRG; STLBL; TABGEN; POINTR	A constant with value = 100000.
MAXBET	116	SBXPRG	Inactive.
MAXBF	115	ENDPRC	Maximum length of "Segments Branched From" (BF) array.
MAXBT	114	SBXPRG; STLBL	Inactive.
MAXSEG	113	ADJUST; BLK2	Maximum length of array in VSTR.
MAXSEG	1	ASSIGN	Maximum length of array in VSTR.
MBETAB	110	ASSIGN; TABGEN; REINIT	Inactive.
MIVTAB	111	ASSIGN, TABGEN, REINIT	Inactive.
MIVTAB	121	TABGEN	Inactive.

Table IV LABEL: BKFOUR (Continued)

Variable Name	QACOM4 Offset	Occurs in Routine(s)	Description
MOVTAB	112	ASSIGN; TABGEN; REINIT	Not used.
MSEG	107	ADJUST; ASSIGN; BLK2	Array of SEGTAB, SEGBT, AND SEGBF origin pointer.
MSEGBF	109	ASSIGN, REINIT, TABGEN, ENDPRC	VSTR pointer for SEGBF origin.
MSEGBT	108	ASSIGN; TABGEN; TAPWRT; REINIT; ENDPRC	VSTR pointer for SEGBT origin.
MSGTAB	107	ASSIGN; TABGEN; REINIT; ENDPRC	VSTR pointer for SEGTAB origin.
NOVTAB	122	TABGEN	Inactive.
NSEG	119	ADJUST; ASSIGN; BLK2	Array of pointers to next available locations of sub-tables SEG, SEGBT, and SEGBF in the VSTR array.
NSEGBF	121	CLENUP, TAPWRT	Next available location in the SEGBF table.
NSEGBT	120	CLENUP; SBXPRG; STLBL; TAPWRT; ENDPRC	Next available location in the SEGBT table.
NSEGBT	129	ADJUST	Next available location in the SEGBT table.
NSGTAB	119	TABGEN, REINIT	Next available location in the SEGTAB table.
NUMSEG	125	ASSIGN; BLK2; CLENUP; TABGEN; REINIT	Absolute segment identification of the current segment.
NUMBR2	102	AUDIT, BLK2; SBXPRG; TABGEN; PCARD1	Second pseudo segment identifier for 'consequent' portion of a logical "IF".
SEGNUM	126	BLK2; DOPROG; SBXPRG; TABGEN; INITSG	Numeric segment number of the user's program.

Table V LABEL: BKZERO

Variable Name	QACOMO Offset	Occurs in Routine(s)	Description
IALL	1	BLK1; TAGBEN; INITAL	Control flag to enable processing of all code numbers.
IBEGIN	2	BLK1; TABGEN; INITAL	Inactive.
IDEBUG	3	AUDIT; BLK1; TABGEN; INITSG; TYPCVT; POINTR; PASS1; INITAL; REINIT; ENDPRC	Debug control flag for printing selected values during execution (= 0 enable, = 1 disable).
KLIST	1204	BLK1; TAGBEN; TAPWRT; INITAL	Inactive.
KPR	1205	BLK1; TABGEN; TAPWRT; INITSG; PASS1; INITAL; PCARD1; REINIT; TERMIN; DOPROG; CLENUP	System print device; value = 06.
KPROG	1207	TABGEN	Inactive.
KRPOG1	1208	TABGEN	Inactive.
KRE	1206	BLK1; TABGEN; TAPWRT; INITAL	System input (reader) device.
LTAPE	1209	BLK1	Inactive.
LTRACK	1210	BLK1	Inactive.
MAPIN	1211	BLK1	Inactive.
MI3PQ	1207	AUDIT; BLK1	Inactive.
NOS	1212	TYPCVT; NINSTR; IFTRAP; PASS1; TRAPIT; DOTERM; INITAL; PCARD1; PCARDN; TYPE	Number of characters in statement label.
NUMBER	1212	BLK1; TABGEN; INITSG	Inactive.

Table V LABEL: BKZERO (Continued)

Variable Name	QACOMO Offset	Occurs in Routine(s)	Description
NUMBR1	1215	AUDIT; BLK1; TABGEN; PCARD1	First pseudo segment identifier for 'predicate' portion of a logical "IF".
NUMSUB	1213	AUDIT; BLK1; TABGEN; INITSG; PASS1	Number of subroutines processed.
STYPE	1214	BLK1; SBXPRG; TABGEN; PCARD1	Identifier for statement type.

2.3.2 Blank "Common"

Variable Name	"EQUIVALENCED" Structure Variable	Occurs in Routine(s)	Description
ISUB	-	AUDIT; TABGEN; REINIT	Number of subroutines processed.
IVAR	IIVAR	GROUP A	Table of "INTEGER" variable names and dimensions.
JTAB	JJTAB	GROUP A	Master array of all specified variables (set in ADDAT).
JVAR	JJVAR	GROUP A	Table of "REAL" variable names and dimensions.
KTAB	KKTAB	AUDIT; TABGEN; INITSG; KTGEN; KT2ND; PCARD1; REINIT	First pass storage array which contains module name and delimiting segment numbers for each module processed.
KVAR	KKVAR	GROUP B	Table of "DOUBLE PRECISION" variable names and dimensions.
LNCNT	-	AUDIT; STNTBL; PCARD1	Line control variables for monitoring number of lines printed per page for page eject.
LVAR	LLVAR	GROUP B	Table of "LOGICAL" variable names and dimensions.
MVAR	MMVAR	GROUP B	Table of "COMPLEX" variable names and dimensions.
NPROG	-	GROUP B	Error summary information; first word contains name of routine, 2-20 contains total number of errors/standard.

GROUP A: FAUDIT, ADEND, ADIOE, ADRY1, ADRY2, ADRY3, ARGU, NTYPE, TYPES, ADERR, ADTRM

GROUP B: Same as GROUP A excluding FAUDIT

2.4 MAJOR SYSTEM CONSTANTS

Constant Name	
IPLUS	'+'
IMINUS	'-'
IASTER	'*'
ISLASH	'/'
ILEFT	'('
IRGHT	')'
IDOLLR	'$'
IBLANK	Hollerith blank
ICOMMA	','
IPEROD	'.'
LALPHA	The 26 letters of the alphabet in hollerith format
LCOMA	','
LLPRN	'('
LRPRN	')'
LBLK	Hollerith blank
LBLANK	Hollerith blank
NUMO	Hollerith zero
NUM	The digits 0 through 9 in hollerith format
NUM9	'9'
MASK	100,000

METRIC SYSTEM

BASE UNITS:

Quantity	Unit	SI Symbol	Formula
length	metre	m	...
mass	kilogram	kg	...
time	second	s	...
electric current	ampere	A	...
thermodynamic temperature	kelvin	K	...
amount of substance	mole	mol	...
luminous intensity	candela	cd	...

SUPPLEMENTARY UNITS:

plane angle	radian	rad	...
solid angle	steradian	sr	...

DERIVED UNITS:

Acceleration	metre per second squared	...	m/s
activity (of a radioactive source)	disintegration per second	...	(disintegration)/s
angular acceleration	radian per second squared	...	rad/s
angular velocity	radian per second	...	rad/s
area	square metre	...	m
density	kilogram per cubic metre	...	kg/m
electric capacitance	farad	F	A·s/V
electrical conductance	siemens	S	A/V
electric field strength	volt per metre	...	V/m
electric inductance	henry	H	V·s/A
electric potential difference	volt	V	W/A
electric resistance	ohm		V/A
electromotive force	volt	V	W/A
energy	joule	J	N·m
entropy	joule per kelvin	...	J/K
force	newton	N	kg·m/s
frequency	hertz	Hz	(cycle)/s
illuminance	lux	lx	lm/m
luminance	candela per square metre	..	cd/m
luminous flux	lumen	lm	cd·sr
magnetic field strength	ampere per metre	...	A/m
magnetic flux	weber	Wb	V·s
magnetic flux density	tesla	T	Wb/m
magnetomotive force	ampere	A	...
power	watt	W	J/s
pressure	pascal	Pa	N/m
quantity of electricity	coulomb	C	A·s
quantity of heat	joule	J	N·m
radiant intensity	watt per steradian	...	W/sr
specific heat	joule per kilogram-kelvin	...	J/kg·K
stress	pascal	Pa	N/m
thermal conductivity	watt per metre-kelvin	...	W/m·K
velocity	metre per second	...	m/s
viscosity, dynamic	pascal-second	...	Pa·s
viscosity, kinematic	square metre per second	...	m/s
voltage	volt	V	W/A
volume	cubic metre	...	m
wavenumber	reciprocal metre	...	(wave)/m
work	joule	J	N·m

SI PREFIXES:

Multiplication Factors	Prefix	SI Symbol
1 000 000 000 000 = 10^{12}	tera	T
1 000 000 000 = 10^{9}	giga	G
1 000 000 = 10^{6}	mega	M
1 000 = 10^{3}	kilo	k
100 = 10^{2}	hecto*	h
10 = 10^{1}	deka*	da
0.1 = 10^{-1}	deci*	d
0.01 = 10^{-2}	centi*	c
0.001 = 10^{-3}	milli	m
0.000 001 = 10^{-6}	micro	μ
0.000 000 001 = 10^{-9}	nano	n
0.000 000 000 001 = 10^{-12}	pico	p
0.000 000 000 000 001 = 10^{-15}	femto	f
0.000 000 000 000 000 001 = 10^{-18}	atto	a

* To be avoided where possible.

U.S. GOVERNMENT PRINTING OFFICE: 1977-714-025/ 98

MISSION
of
Rome Air Development Center

RADC plans and conducts research, exploratory and advanced development programs in command, control, and communications (C^3) activities, and in the C^3 areas of information sciences and intelligence. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.